Volume: 03, Issue: 01, Pages:1025-1030 (2011)

Processing Reliability based a Clever Task Allocation Algorithm to Enhance the Performance of Distributed Computing Environment

Dr. Kapil Govil
Department of Computer Applications, Teerthanker Mahaveer University, Moradabad-244001
Email: drkapilgovil@gmail.com

ABSTRACT

Distributed Computing Environment (DCE) is one the appropriate network for providing the optimal solution in real-time applications problems. The systematic allocation of tasks needs to plays the key role to optimize the overall processing reliability of the DCE. DCE consists of multiple autonomous computers that communicate through a common communication channel. The computers interact with each other in order to achieve a common goal. A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs. The allocation problems in any computer system play the key role for deciding the performance of the system. The allocation of tasks in DCE has an important role to the evaluation of the performance of the network. The problem of processing of "m" tasks to "n" processors (m > n) in a distributed networks is addressed here through a new modified tasks allocation policy for the task processing in a DCE. The model, presented in this paper allocates the tasks to the processor to enhance the performance of the DCE.

Keywords - **Allocation, Distributed Computing Environment, Performance, Task.**

Date of Submission: March 21, 2011 Date of Acceptance: May 03, 2011

I. INTRODUCTION

Distributed Computing Environment (DCE), utilizes a network of many computers, each accomplishing a portion of an overall task, to achieve a computational result much more quickly than with a single computer. In addition to a higher level of computing power, DCE also allows many users to interact and connect openly. Different forms of distributed computing allow for different levels of openness, with most people accepting that a higher degree of openness in a DCE is beneficial. The main research problem for such networks is the allocation problem, in which processing reliability is to be maximized of the DCE.

One of the major research problems for DCE is the allocation problem, in which tasks are assigned to various processors of the network, in such a way that processing reliability is to be maximized as per the requirement. These problems may be categorized as static ([1], [2], [3], [4]) and dynamic ([4], [5], [6], [7], [8]) types of task allocation. Some of the other related methods have been reported in the literature, such as, Integer Programming ([9], [10], [11], [12], [13], [14]), Load Balancing ([5], [15], [16], [17], [18], [19], [20], [21]), Modeling ([22], [23], [24]) and Reliability Computation ([25], [26]). Tasks are allocated to various processors of the distributed network in such a way that overall processing reliability of the network should be maximized. As it is well known that the tasks are more than the number of processors of the network.

II. OBJECTIVE

In the Distributed Computing Environment (DCE), it is the common problem to allocate tasks where the number of tasks is more than the number of processors. In the present research paper the type of allocation of task to the processor is static in nature. The objective of the present research paper is to enhance the performance of the DCE by using the proper utilization of its processors and as well as proper allocation of tasks. The present research paper reduces the overall processing reliability for the environment of DCE through proper allocation of tasks to the appropriate processor.

III. NOTATIONS

p	Processor
t	Task
n	Number of Processors
m	Number of Tasks
HCT	Highest Communication Matrix

RTPM Revised Task Processor Matrix
TAT Task Allocation Table
TCM Task Communication Matrix
TPM Task Processor Matrix

IV. TECHNIQUE

To evaluate the overall optimal processing reliability for the environment of DCE we have chosen the problem where a Set P = {p_1, p_2, p_3, … p_n} of 'n' processors with dissimilar configuration and a Set T = {t_1, t_2, t_3, t_4, …t_m} of 'm' tasks where m> n. First of all we read the processing reliability of tasks on processors in TPM(,) of order n * m and Task Communication Matrix in TCM(,) of order m*m. Then, compute the sum of processing reliability and average of processing reliability in TPM(,), and store the results in TPM_1(,). After that eliminate those task-processor combinations which have more processing reliability than average processing reliability until each column contain similar type of processing type, and store the results in Revised Task Processor Matrix namely RTPM(,) of order n * m. Then find the communication reliability from TCM(,) among other tasks for each and every task in the descending order, and mention in Highest Communication Table (HCT) and allocate possible tasks from HCT to the processors where tasks can be get processed on the same processor by using RTPM(,), and store the allocations in the Task Allocation Table (TAT). Finally we have to show the Optimal Results.

V. ALGORITHM

Step1: Read the number of tasks in m

Step2: Read the number of processor in n

Step3: Read the processing reliability of tasks on processors in TPM(,) of order n * m

Step4: Read the communication matrix in TCM(,) of order m*m

Step5: Compute the sum of processing reliability and average of processing reliability in TPM(,), and store the results in TPM_1(,).

Step6: Eliminate those task-processor combinations which have more processing reliability than average processing reliability until each column contain similar type of processing type, and store the results in Revised Task Processor Matrix namely RTPM(,) of order n * m.

Step7: Find the communication reliability from TCM(,) among other tasks for each and every task in the descending order, and mention in Highest Communication Table (HCT).

Step8: Allocate possible tasks from HCT to the processors where tasks can be getting processed on the same processor by using RTPM (,) until all tasks get allocated, and store the allocations in the Task Allocation Table (TAT).

Step9: Show the Optimal Results

VI. IMPLEMENTATION

In the present research paper, we have chosen an environment of distributed processing having a set P of 4 processors {p_1, p_2, p_3, p_4} and a set T of 12 tasks {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}}. This problem is shown in the figure 1. The processing reliability (r) on various processors are mentioned in Task Processor Matrix namely TPM (,) of order 4 * 12.

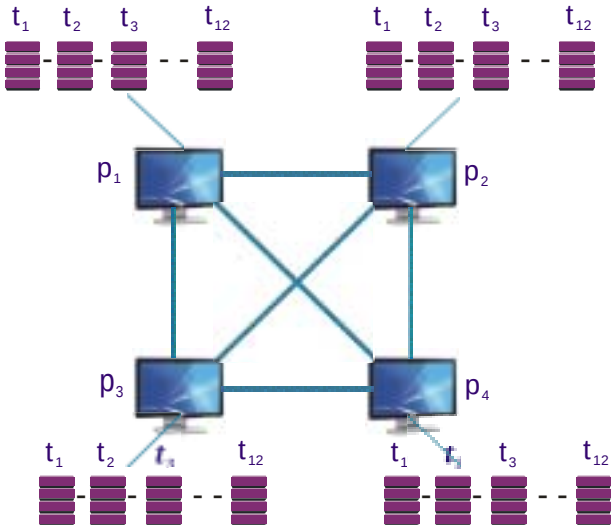

Fig. 1. Task allocation problem in distributed computing environment

TPM(,) =

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
p_1	0.998276	*	0.998396	0.998327	0.998765	*	*	0.997263	0.993256	0.998376	0.998532	0.998674
p_2	0.998927	0.999412	0.991234	0.998436	0.992769	0.992345	*	*	0.998123	0.998873	0.998883	0.997387
p_3	0.997234	0.991040	0.999876	0.998437	*	0.999123	0.998777	*	0.998411	*	0.998812	*
p_4	*	0.991735	*	*	0.992096	0.998736	*	*	0.998325	*	0.998532	0.998623

The communication amongst the tasks has also taken into consideration. Its matrix representation has been given by the asymmetric Task Communication Matrix namely TCM (,) of order 12 * 12.

TCM(,) =

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
t_1	0	3	8	4	1	5	3	8	6	2	6	8
t_2	5	0	6	5	9	4	9	3	1	8	9	7
t_3	2	4	0	6	3	3	4	5	2	1	5	4
t_4	1	9	8	0	7	2	1	4	4	5	1	3
t_5	5	8	6	4	0	1	6	9	8	4	9	2
t_6	1	2	3	4	5	0	2	7	9	7	8	6
t_7	5	6	2	9	8	7	0	5	5	6	3	4
t_8	6	4	1	9	8	2	1	0	5	3	4	1
t_9	2	5	8	7	6	3	4	1	0	2	3	5
t_{10}	8	4	2	3	2	8	4	6	5	0	4	7
t_{11}	5	7	6	4	2	4	2	5	5	5	0	3
t_{12}	1	2	6	5	7	3	5	8	8	9	2	0

Now, we evaluate the sum of processing reliability and average of processing reliability in TPM(,), which is mentioned in TPM_1(,).

TPM_1(,)=

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
p_1	0.998276	*	0.998396	0.998327	0.998765	*	*	0.997263	0.993256	0.998376	0.998532	0.998674
p_2	0.998927	0.999412	0.991234	0.998436	0.992769	0.992345	*	*	0.998123	0.998873	0.998883	0.997387
p_3	0.997234	0.991040	0.999876	0.998437	*	0.999123	0.998777	*	0.998411	*	0.998812	*
p_4	*	0.991735	*	*	0.992096	0.998736	*	*	0.998325	*	0.998532	0.998623
Σ	2.994437	2.982187	2.989506	2.995200	2.983630	2.990204	0.998777	0.997263	3.988115	1.997249	3.994767	2.994684
Avg.	0.998145	0.994062	0.996502	0.998400	0.994543	0.996734	0.998777	0.997263	0.997028	0.998624	0.998691	0.998228

On eliminating those task – processor combinations which have more processing reliability than average processing reliability until each column contain similar type of processing type, we store the results in Revised Task Processor Matrix namely RTPM(,) of order 4 * 12.

RTPM(,) =

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
p_1	0.998276	*	0.998396	*	0.998765	*	*	0.997263	0.993256	*	*	0.998674
p_2	0.998927	0.999412	*	0.998436	*	*	*	*	0.998123	0.998873	0.998883	*
p_3	*	*	0.999876	0.998437	*	0.999123	0.998777	*	0.998411	*	0.998812	*
p_4	*	*	*	*	*	0.998736	*	*	0.998325	*	*	0.998623

From, TCM(,) we find the communication reliability among other tasks for each and every task in the descending order, and mention in Highest Communication Table (HCT).

Volume: 03, Issue: 01, Pages:1025-1030 (2011)

TABLE I. HIGHEST COMMUNICATION TABLE

Tasks	Communication reliability
t_1	$t_3, t_8, t_{12}, t_9, t_{11}, t_6, t_4, t_2, t_7, t_{10}, t_5$
t_2	$t_5, t_7, t_{11}, t_{10}, t_{12}, t_3, t_1, t_4, t_6, t_8, t_9$
t_3	$t_4, t_8, t_{11}, t_2, t_7, t_{12}, t_5, t_6, t_1, t_9, t_{10},$
t_4	$t_2, t_3, t_5, t_{10}, t_8, t_9, t_{12}, t_6, t_1, t_7, t_{11}$
t_5	$t_8, t_{11}, t_2, t_9, t_3, t_7, t_1, t_4, t_{10}, t_{12}, t_6$
t_6	$t_9, t_{11}, t_8, t_{10}, t_{12}, t_5, t_4, t_3, t_2, t_7, t_1$
t_7	$t_4, t_5, t_6, t_2, t_{10}, t_1, t_8, t_9, t_{12}, t_{11}, t_3$
t_8	$t_4, t_5, t_1, t_9, t_2, t_{11}, t_{10}, t_6, t_3, t_7, t_{12}$
t_9	$t_3, t_4, t_5, t_2, t_{12}, t_7, t_6, t_{11}, t_1, t_{10}, t_8$
t_{10}	$t_1, t_6, t_{12}, t_8, t_9, t_2, t_7, t_{11}, t_4, t_3, t_5$
t_{11}	$t_2, t_3, t_1, t_8, t_9, t_{10}, t_4, t_6, t_{12}, t_5, t_7$
t_{12}	$t_{10}, t_8, t_9, t_5, t_3, t_4, t_7, t_6, t_2, t_{11}, t_1$

On allocating possible tasks from HCT to the processors where tasks can be get processed on the same processor by using RTPM(,), and store the allocations in the Task Allocation Table (TAT).

TABLE II. TASK ALLOCATION TABLE

Processor	Allocated tasks	Optimal Reliability
p_1	t_1*t_3	0.996674
p_2	t_2*t_{11}	0.998295
p_3	t_4*t_9	0.996850
p_4	t_6*t_{12}	0.997360

TABLE III. TASK ALLOCATION TABLE

Processor	Allocated tasks	Optimal Reliability
p_1	t_5*t_8	0.996031
p_2	t_{10}	0.998873
p_3	t_7	0.998777
p_4	-	-

The graphical representation of the optimal allocation is shown in figure 2.

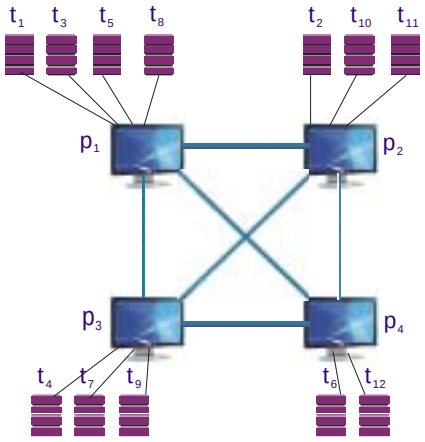

Fig. 2. Allocated tasks to the processors

VII. CONCLUSION

In this research paper we have chosen the problem, in which the number of the tasks is more than the number of processors of the DCE. The model mentioned in this paper is based on the consideration of processing reliability of the tasks to various processors. The method is presented in pseudo code and implemented on the several sets of input data to test the performance and effectiveness of the pseudo code. It is the common requirement for any allocation problem that the tasks have to be processed with maximum reliability. Here, performance is measured in terms of processing reliability of the task that has been processed by the processors of the network and also these tasks have been processed optimally. The overall optimal results of the example are mentioned in the Optimal Result Table.

TABLE IV. OPTIMAL RESULT TABLE

Processor	Allocated tasks	Optimal Result
p_1	$t_1*t_3*t_5*t_8$	0.982981
p_2	$t_2*t_{10}*t_{11}$	
p_3	$t_4*t_7*t_9$	
p_4	t_6*t_{12}	

As we know that, the analysis of an algorithm is mainly focuses on time complexity. Time complexity is a function of input size 'n'. It is referred to as the amount of time required by an algorithm to run to completion. The time complexity of the above mentioned algorithm is $O(m^2n^2)$. By taking several input examples, the above algorithm returns following results,

TABLE V. COMPLEXITY TABLE

No. of processors (n)	No. of tasks (m)	Optimal Results
3	4	144
3	5	225
3	6	324
3	7	441
3	8	576
4	5	400
4	6	576

4	7	784
4	8	1024
4	9	1296
5	6	900
5	7	1225
5	8	1600
5	9	2025
5	10	2500

The graphical representation of the above results are shown by figure 3, 4 and 5 as mentioned below,

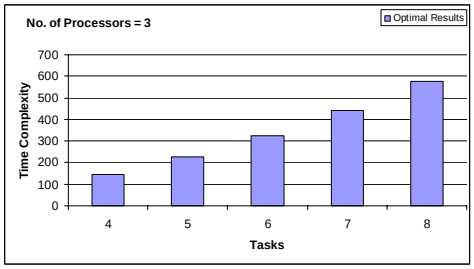

Fig. 3. Optimal Result

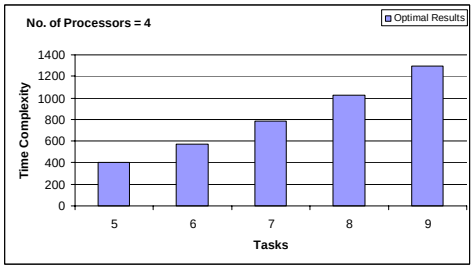

Fig. 4. Optimal Result

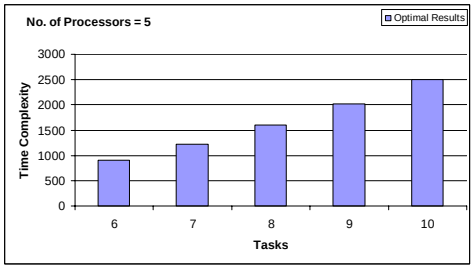

Fig. 5. Optimal Result

REFERENCES

[1]. Avanish Kumar, An Algorithm for Optimal Index to Tasks Allocation Based on Reliability and cost, *Proc. of 'International Conference on Mathematical Modeling*, Roorkee, 2001, 150-155.

[2]. A. Kumar, M. P. Singh and P. K. Yadav, An Efficient Algorithm for Allocating Tasks to Processors in a Distributed System, *Proc. of the '19th National System Conference, SSI*, Coimbatore, 1995, 82-87.

[3]. A. Kumar, M. P. Singh, and P. K. Yadav, A Fast Algorithm for Allocating Tasks in Distributed Processing System, *Proc. of the '30th Annual Convention of CSI*, Hyderabad, 1995, 347-358.

[4]. Yu-Kwong Kwok, Anthony A. Maciejewski, Howard Jay Siegel, Ishfaq Ahmad, and Arif Ghafoor, A semi-static approach to mapping dynamic iterative tasks onto heterogeneous computing systems. *Elsevier Inc*., 66(1), 2006, 77-98.

[5]. Jacques Bahi, Raphael Couturier, and Flavien Vernier, Synchronous distributed load balancing on dynamic networks. *Elsevier Inc*., 65(11), 2005, 1397-1405.

[6]. Avanish Kumar, Optimizing for the Dynamic Task Allocation, *Proc. of the 'III Conference of the International Academy of Physical Sciences*, Allahabad, 1999, 281-294.

[7]. A. Kumar, M. P. Singh, and P. K. Yadav, An Efficient Algorithm for Multi- processor Scheduling with Dynamic Reassignment, *Proc. of the '6th National seminar on theoretical Computer Science*, Banasthally Vidyapeeth, 1996, 105-118.

[8]. J. Palmer, and I. Mitrani, Optimal and heuristic policies for dynamic server allocation. *Elsevier Inc*., 65(10), 2005, 1204-1211.

[9]. Jeffery L. Kennington, Eli V. Olinick, Gheorghe Spiride, Basic mathematical programming models for capacity allocation in mesh-based survivable networks, *Omega*, 35(6), December 2007, 629-644.

[10]. I Kuban Altınel, Necati Aras, Evren Güney, Cem Ersoy, Binary integer programming formulation and heuristics for differentiated coverage in heterogeneous sensor networks, *Computer Networks*, 52(12), August 2008, 2419-2431.

[11]. Brian Ensink, Joel Stanley, and Vikram Adve, Program Control Language: a programming language for adaptive distributed applications, *Elsevier Inc.*, 63(12), 2003, 1082 –1104.

[12]. O. I. Dessoukiu-EI, and W. H. Huna, Distributed Enumeration on Network Computers, *IEEE Transactions on Computer*, 29, 1980, 818-825.

[13]. K B. Misra, and U. Sharma, An Efficient Algorithm to solve Integer Programming Problem arising in System Reliability Design, *IEEE Transactions on Reliability*, 40, 1991, 81-91.

[14]. Muhammad K. Dhodhi, Imtiaz Ahmad, Anwar Yatama and Ishfaq Ahmad. An Integrated Technique for Task Matching and Scheduling onto Distributed Heterogeneous Computing Systems, *Elsevier Inc.*, 62(9), 2002, 1338 – 1361.

[15]. Andrey G. Bronevich, Wolfgang Meyer, Load balancing algorithms based on gradient methods and their analysis through algebraic graph theory, *Journal of Parallel and Distributed Computing*, 68(2), February 2008, 209-220,.

[16]. Bruce Hendrickson, Karen Devine, Dynamic load balancing in computational mechanics,

Computer Methods in Applied Mechanics and Engineering, 184(2-4), April 2000, 485-500.

[17]. Maria Joao Alves, Joao Clímaco, A review of interactive methods for multiobjective integer and mixed-integer programming, *European Journal of Operational Research*, 180(1), July 2007, 99-115.

[18]. Daniel Grosu, and Anthony T. Chronopoulos, Noncooperative load balancing in distributed systems, *Elsevier Inc*., 65(9), 2005, 1022-1034.

[19]. Dorta, C. Leon, C. Rodríguez, Performance analysis of Branch-and-Bound skeletons, *Mathematical and Computer Modeling*, 51(3-4), February 2010, 300-308.

[20]. Saeed Iqbal, and Graham F. Carey, Performance analysis of dynamic load balancing algorithms with variable number of processors. *Elsevier Inc*., 65(8), 2005, 934-948.

[21]. Shu Wanneng, Wang Jiangqing, Min-Min Chromosome, Genetic Algorithm for Load Balancing in Grid Computing, *International Journal of Distributed sensor network*, 5(1), 2009, 62-63.

[22]. Javier Contreras, Arturo Losi, Mario Russo, and Felix F. Wu, DistOpt: A Software Framework for Modeling and Evaluating Optimization Problem Solutions in Distributed Environments, *Elsevier Inc*., 60(6), 2000, 741 – 763.

[23]. Kent Fitzgerald, Shahram Latifi, and Pradip K. Srimani, Reliability Modeling and Assessment of the Star-Graph Networks, *IEEE Transactions on Reliability*, 51, 2002, 49-57.

[24]. Rene L. Bierbaum, Thomas D. Brown, and Thomas J. Kerschen, Model-Based Reliability Analysis, *IEEE Transactions on Reliability*, 51, 2002, 133-140.

[25]. Gamal Attiya, Yskandar Hama, Task allocation for maximizing reliability of distributed systems: A simulated annealing approach, *Journal of Parallel and Distributed Computing*, 66(10), October 2006, 1259-1266.

[26]. Bo Yang, Huajun Hu, Suchang Guo, Cost-oriented task allocation and hardware redundancy policies in heterogeneous distributed computing systems considering software reliability, *Computers & Industrial Engineering*, 56(4), May 2009, 1687-1696.